Attorney Docket No. CISCP531C1
Client Ref. No. 9218

APPLICATION FOR UNITED STATES PATENT

ROUTER SOFTWARE UPGRADE EMPLOYING REDUNDANT PROCESSORS

By Inventors:

KENNETH MOBERG
1105 Pinecrest Drive
Boulder Creek, CA 95006
Citizen of the United States of America

RONNIE KON
113 Albert Court
Los Gatos, CA 95032
Citizen of the United States of America

Assignee: CISCO TECHNOLOGY, INC.
170 W. Tasman Drive
San Jose, CA 95134

Entity Status: Large

RITTER, LANG & KAPLAN LLP
12930 Saratoga Avenue, Suite D1
Saratoga, California 95070
(408) 446-8690

ROUTER SOFTWARE UPGRADE EMPLOYING REDUNDANT PROCESSORS

STATEMENT OF RELATED APPLICATIONS

The present invention is a continuation of U.S. Patent Application No. 09/512,990, filed February 24, 2000, and related to the subject matter of U.S. Patent Application No. 09/205,577, filed on December 4, 1998 and U.S. Patent Application No. 09/205,554, filed on December 4, 1998. The contents of each applications are herein incorporated by reference for all purposes in their entirety.

BACKGROUND OF THE INVENTION

The present invention relates to computer networks and more particularly to systems and methods for upgrading packet processing software without interfering with packet flow.

Routers are internetworking devices that are typically used to connect similar and heterogeneous network segments into internetworks. A typical router includes various interfaces that send and receive packets. The router receives a given packet through a first interface, processes the packet to determine how to best forward the packet to its destination, and then based on this determination transmits the packet through a selected second interface.

In a high speed data communication network, a router will handle a very large and continuous flow of packets. Any sustained interruption of router operation may adversely affect network performance due to increased latency or even loss of packets.

Many popular types of routers operate under the control of packet processing software. The packet processing software directly manipulates the individual packets to be forwarded by the router. As with any software controlled device, it is sometimes desirable or necessary to replace or upgrade the software. A problem arises in that changing software requires halting packet processing because it is the software that directly manipulates the packet. Systems and methods for replacing router software while minimizing impact on network operation are needed.

SUMMARY OF THE INVENTION

Systems and methods for replacing software controlling active routers while minimizing impact on network operation are provided by virtue of one embodiment of the present invention. The software replacement process takes advantage of packet processor redundancy. In one embodiment, an active packet switching device to be reprogrammed is de-activated and a redundant packet switching device takes over. The no longer active packet switching device is reprogrammed before being reactivated.

A first aspect of the present invention provides a computer-implemented method for replacing an active packet switching device without interfering with packet flow. The method includes receiving a failover message at a currently active packet switching device (A), transferring protocol state information from the packet switching device (A) to a currently inactive packet switching device (B), de-activating the packet switching device (A) and activating the packet switching device (B).

A second aspect of the present invention provides a computer-implemented method for reprogramming an active packet switching device without interfering with packet flow. The method includes: de-activating a current packet switching device (A) and activating a standby packet switching device (B) to handle packet flow previously handled by the packet switching device (B), thereafter reprogramming the packet switching device (A), and thereafter de-activating the packet switching device (B) and re-activating the packet switching device (B).

A further understanding of the nature and advantages of the inventions herein may be realized by reference to the remaining portions of the specification and the attached drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 depicts a router including redundant packet processing capabilities according to one embodiment of the present invention.

Fig. 2 depicts a top level flowchart describing steps of upgrading packet processing software according to one embodiment of the present invention.

Fig. 3 depicts a detailed flowchart describing steps of upgrading packet processing software according to one embodiment of the present invention.

DESCRIPTION OF SPECIFIC EMBODIMENTS

One embodiment of the present invention is directed toward upgrading packet processing software while minimizing or eliminating any interruptions in the handling of packets. This is preferably done by providing at least two independent packet processors with access to the same interfaces so that while one packet processor is being upgraded, the other packet processor can take over packet processing operations. The handoff between packet processors is fast and transparent to overall network operation.

Generally, the packet processing software management techniques of the present invention may be implemented in software and/or hardware. For example, they can be implemented in an operating system kernel, in separate user processes, in a library package bound into network applications, on a specially constructed machine, or on a network interface card. In a specific embodiment of this invention, these techniques may be implemented in software such as an operating system or in an application running on an operating system.

A software or software/hardware hybrid packet processing management system of this invention is preferably implemented on a general-purpose programmable machine selectively activated or reconfigured by a computer program stored in memory. Such a programmable machine may be a network device designed to handle network traffic. Such network devices typically have multiple network interfaces including frame relay and ISDN interfaces, for example. Specific examples of such network devices include

routers and switches. For example, the packet processing management systems of this invention may operate on specially configured routers such as those available from Cisco Systems, Inc. of San Jose, California. A general architecture for some of these machines will appear from the description given below. In an alternative embodiment, the packet processing management system may be implemented on a general-purpose network host machine such as a personal computer or workstation adapted to interface with computer networks.

Referring now to FIG. 1, a router 10 suitable for implementing the present invention includes interfaces 68, and a bus 15 (e.g., a PCI bus). Router 10 incorporates redundant packet processing capabilities and therefore includes two CPUs 61A and 61B. As shown, CPU 61A includes a memory 62A and a processor 63A. Similarly, CPU 61B includes a memory 62B and a processor 63B. At any one time, one of CPU 61A and CPU 61B is the "primary" packet processor responsible for handling and forwarding newly received packets. When acting under the control of appropriate software or firmware, CPUs 61A and 61B are responsible for such router tasks as routing table computations, network management, and general processing of packets. It preferably accomplishes all these functions under the control of software including an operating system (e.g., a version of the Internetwork Operating System (IOS®) of Cisco Systems, Inc.) and any appropriate applications software. Processors 63A and 63B may be, e.g., microprocessors of the Motorola family or microprocessors of the MIPS family of microprocessors. In an alternative embodiment, processors 63A and 63B are specially designed hardware for controlling the operations of router 10. Memories 62 A and 62B

can be non-volatile RAM and/or ROM. However, there are many different ways in which memory could be coupled to the system.

The interfaces 68 are typically provided as interface cards (sometimes referred to as "line cards"). Generally, they control the sending and receiving of data packets over the network and sometimes support other peripherals used with the router 10. Among the interfaces that may be provided are Ethernet interfaces, frame relay interfaces, cable interfaces, DSL interfaces, token ring interfaces, and the like. In addition, various high-speed interfaces may be provided such as fast Ethernet interfaces, Gigabit Ethernet interfaces, ATM interfaces, HSSI interfaces, POS interfaces, FDDI interfaces and the like. Generally, these interfaces may include ports appropriate for communication with the appropriate media. In some cases, they may also include an independent processor and, in some instances, volatile RAM. The independent processor may control such communications intensive tasks as packet switching, media control, and management. By providing separate processors for the communications intensive tasks, these interfaces allow the CPUs 61A and 61B to efficiently perform routing computations, network diagnostics, security functions, etc. Router 10 may further include a packet memory 72 for intermediate storage of packets being forwarded by router 10.

Although the system shown in FIG. 1 is one specific router of the present invention, it is by no means the only architecture on which the present invention can be implemented. For example, an architecture having a single processor that handles communications as well as routing computations, etc. is often used. Further, other types of interfaces and media could also be used with the router.

Regardless of a network device's configuration, it may employ one or more memories or memory modules (including memories 62A and 62B) configured to store program instructions for the general-purpose network operations and packet processing and management functions described herein. The program instructions may control the operation of an operating system and/or one or more applications, for example.

Because such information and program instructions may be employed to implement the systems/methods described herein, the present invention relates to machine-readable media that include program instructions, state information, etc. for performing various operations described herein. Examples of machine-readable media include, but are not limited to, magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks; magneto-optical media such as optical disks; and hardware devices that are specially configured to store and perform program instructions, such as read-only memory devices (ROM) and random access memory (RAM). The invention may also be embodied in a carrier wave travelling over an appropriate medium such as airwaves, optical lines, electric lines, etc. Examples of program instructions include both machine code, such as produced by a compiler, and files containing higher level code that may be executed by the computer using an interpreter.

Fig. 2 is a flowchart generally describing steps of upgrading packet processing software in a system such as router 10 that employs redundant CPUs for performing packet processing operations. It is desired to upgrade the packet processing software of CPU 61A. It is assumed that CPU 61A is currently the primary processor and is handling

packets received via interfaces 68 and performing operations on packets stored in packet memory 72. CPU 61B is a secondary processor that is available as a reserve in the event of a failure by CPU 61A. At step 202, CPU 61A hands over current packet processing responsibility to CPU 61B so that CPU 61A is no longer the primary packet processor. Now, CPU 61B is the primary packet processor. CPU 61A is now a secondary processor. While CPU 61A is secondary processor, at step 204, its software may be upgraded by writing to memory 62A. Once the software of CPU 61A has been upgraded, it can again assume primary responsibility for processing packets. CPU 61B transfers responsibility back to CPU 61A at step 206 so that CPU 61A is again the primary processor and CPU 63B is the secondary processor. CPU 61A then resumes processing packets using its new software. According to the present invention, an entire upgrading operation may be accomplished without any dropped packets as will be explained below. In an alternative embodiment, CPU 61B is upgraded to new software before the transfer of responsibility while it is operating as the secondary processor.

Fig. 3 is a flowchart describing detailed steps of upgrading packet processing software according to one embodiment of the present invention. At step 302, CPU 61A which is now the primary packet processor receives a remotely generated message telling it to upgrade its software. The message may be generated at a remote network management workstation. The message may include a URL indicating the location of the new software to download and install. CPU 61A, in response, begins a failover process to shift packet processing responsibility to CPU 61B. This process is similar to what would be followed in the event of a failure by CPU 61A.

At step 304, CPU 61A stops processing new data packets and allows them to queue up within packet memory 72. At step 306, CPU 61A sends protocol data to CPU 61B to facilitate the handover. The protocol information to be transferred may include, e.g., routing tables, negotiated state information including information as to which links are being ignored because of spanning tree negotiation, authorization status for remote point to point links, and other learned routing state information, etc. There are various ways that the protocol data may be passed between CPUs 61A and 61B. For example, there may be a dedicated serial port connection between the CPUs. Alternatively, CPU 61A may place the protocol data in a packet buffer within packet memory 72 that CPU 61B can then read. Once the protocol data has been passed, CPU 61A sends a further message to CPU 61B telling CPU 61B to begin acting as the primary packet processor at step 308. At step 310, CPU 61B begins receiving and processing packets.

At step 312, CPU 61A loads new software from the location specified in the software update message of step 302. The new software may be, for example, a new version of IOS. At step 314, CPU 61A begins operating as the secondary packet processor. CPU 61A can become the secondary packet processor by rebooting as a secondary for example. Operations then may continue for some time with CPU 61B operating as the primary packet processor and CPU 61A operating as a secondary packet processor.

In order to resume primary operations at CPU 61A with the new software, a failover message is received by CPU 61B from any network node. At step 318, CPU 61B stops processing new packets. Then at step 320, CPU 61B sends the necessary

protocol information back to CPU 61A as was done at step 306. Then, at step 322 CPU 61B sends a message to CPU 61A directing CPU 61A to resume operation as primary packet processor. CPU 61B can then begin running as secondary or reboot to be the secondary packet processor.

Each protocol operating on CPU 61A may send its own data message to its corresponding protocol entity on CPU 61B. In this way, the protocols may operate uninterrupted around the failover. The handovers should preferably occur within 1 or 2 seconds to avoid dropping packets.

It is understood that the examples and embodiments described herein are for illustrative purposes only and that various modifications and changes in light thereof will be suggested to persons skilled in the art and are to be included in the spirit and purview of this application and scope of the appended claims and their full scope of equivalents. All publications, patents, and patent applications cited herein are hereby incorporated by reference.